PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/445, 9/44

A1

(11) International Publication Number: WO 99/23556

(43) International Publication Date: 14 May 1999 (14.05.99)

(21) International Application Number: PCT/SE98/01943

(22) International Filing Date: 27 October 1998 (27.10.98)

(30) Priority Data:
08/961,446 30 October 1997 (30.10.97) US

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S–126 25 Stockholm (SE).

(72) Inventor: RÖJESTÅL, Hans, Jonas, Peter; Ringblomman 14, S–178 52 Ekerö (SE).

(74) Agent: ERICSSON RADIO SYSTEMS AB; Common Patent Dept., S–164 80 Stockholm (SE).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: REMOTE SOFTWARE DOWNLOAD WITH AUTOMATIC ADJUSTMENT FOR DATA ACCESS COMPATIBILITY

(57) Abstract

In a method of replacing a first version of software with a second version of software, the second version of software is loaded into a data processor on which the first version of software is running. While the first version of software is running on the data processor, the first version of software is used to modify a data memory for compatibility with the second version of software.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

REMOTE SOFTWARE DOWNLOAD WITH AUTOMATIC ADJUSTMENT FOR DATA ACCESS COMPATIBILITY

FIELD OF THE INVENTION

This invention relates generally to downloading software to a program memory from a remote source and, more particularly, to such a remote software download with facilities to automatically adjust for data access compatibility.

BACKGROUND OF THE INVENTION

In the field of data processing, software designers continually develop and release new versions of software having improved functionality relative to older versions of the software. In conventional computer systems, there are various reasons for substituting a newer version of software for an older version of software, and for substituting an older version of software for a newer version of software.

For example, when a new version of software includes improved functionality and/or improved diagnostic capabilities relative to an older version of the software, it is often desirable to replace the older version of software with the newer, improved version of software. As another example, if a spare or replacement component that has already been programmed with an older version of software is introduced into a system whose other components utilize a newer version of the software, then it may be desirable to modify the spare component to replace its older software version with the newer software version used in the other system components so that, for example, all system components are executing the same version of software.

In other instances, it may be desirable to replace a newer version of software with an older version of software. For example, if a new component that has already been programmed with a new version of software is introduced into an older system whose other components run an older version of the software, it may be desirable to modify the new component by replacing its newer version of software with the older version of software used in the other system components to, for example, ensure that all system components are executing the same version of software. As another example, if one component of a system is updated to run a newer version of software,

and thereafter the system develops operational problems, then it may be desirable to change the software in that one component back to the older version that was in use before the operational problems arose.

In a distributed computer system wherein the various components can be geographically located quite remotely from one another, it is often desirable, when substituting one version of software for another version, to download the desired version of software from a remote source to a program memory in the desired destination. The download is typically executed via a connection that is otherwise used for other types of communication between the remote source and the data processor. For example, the remote source could be another data processor in the system, and the connection could be a communication bus connected between the data processors. This is illustrated in FIGURE 1 wherein remote source 11 is connected by communication bus 23 to data processor 13 and performs a remote software download via bus 23 into program memory 15 of data processor 13.

FIGURE 2 illustrates the effect of the remote software download of FIGURE 1. More specifically, the currently executing software SWc is executed by execution portion 21 of data processor 13 until the downloaded software SWd is downloaded from the remote source 11, at which time SWd begins to be executed by execution portion 21. Because the system already provides bus 23 for communication between the remote source 11 and data processor 13, and because the geographic location of data processor 13 may make it difficult to physically or manually access data processor 13, the bus 23 is advantageously used to perform the remote software download to data processor 13.

FIGURES 3 and 4 respectively illustrate relevant portions of SWc and SWd in diagrammatic fashion. SWc and SWd are different versions of software. SWc includes a data handler portion 31 which handles data from data memory 17 of FIGURE 1 differently than does the data handler portion 41 of SWd in FIGURE 4. Thus, the data accessed at 33 by data handler 31 of SWc in FIGURE 3 will not necessarily be compatible with the data handler 41 of SWd in FIGURE 4, and the data accessed at 43 by data handler 41 of SWd will not necessarily be compatible with the data handler 31 of SWc in FIGURE 3. For example, each of the different data handlers 31 and 41 of respective software versions SWc and SWd may require a

different data format than the other one and/or may require a different data memory layout (or configuration) than the other one.

The accessed data in FIGURES 3 and 4 can be, for example, a database, a volatile memory such as a RAM, a non-volatile memory such as flash memory, floppy disk or hard disk, or a file format. As indicated above, SWc may require the accessed data to have a different format than is required by SWd, and SWc may require the data memory 17 to be configured differently than the memory configuration required by SWd. Thus, replacing SWc with SWd by remote download as illustrated in FIGURES 1 and 2 will often result in SWd attempting to access a data memory 17 whose data format and memory configuration is not compatible with that required by SWd.

One conventional approach to this problem is to design the original software version with a data handler that provides all possible data format and memory configuration parameters that could be needed by future versions of the software. In this manner, any version of the software would, when downloaded to replace another version, find data memory 17 having a data format and memory configuration that it requires. This is of course a very difficult task because it is virtually impossible to predict the data format and memory configuration that will be required by future versions of the software. Even if it were possible to perform this task, it would be extremely expensive to perform because of the extended amount of time necessary to predict all possible data formats and memory configurations that could be needed in the future.

Another conventional solution to the data access compatibility problem is visiting the site of data memory 17 and manually reformatting the data and reconfiguring the memory at the time that SWd is downloaded. Such manual intervention is very costly and time consuming (data memory 17 may be very remotely located), and may cause major disturbances in data flow to and from the other system components illustrated at 19 in FIGURE 1.

Another conventional solution recognizes that the data handler of a newer version of software can, in most cases, handle the data format and memory configuration required by the data handlers of all previously released versions of the software. This solution is illustrated by the flow charts of FIGURES 5 and 6.

FIGURE 5 illustrates the download control of SWc. Recall from FIGURE 2 that SWc is executing on data processor 13 until SWd is downloaded from remote source 11. The download control portion 50 of SWc receives and stores the SWd download at 51, and awaits at 53 a command from the remote source 11 to allow SWd to begin execution. When the remote source commands execution of SWd, SWc at 55 causes data processor 13 to restart, running SWd.

FIGURE 6 illustrates the download reformat portion 60 of SWd. At 61, SWd determines whether or not it supports the current database (DB) format, which in most cases it will because it is a newer version of software than SWc. Nevertheless, if SWd does not support the current database format, then SWd, which is capable of handling the SWc format of the database, simply reformats the database to its preferred format at 63. At 65, SWd determines whether or not it supports the current memory configuration, which in most cases it will because it is a newer version of software than SWc. Nevertheless, if SWd does not support the current memory configuration, it uses its capability to handle the SWc memory configuration and simply changes the configuration to its own preferred configuration at 67.

It is again emphasized that the technique, illustrated in FIGURES 5 and 6, of downloading a newer version of software and having that newer version reformat the data and reconfigure the memory as needed, only addresses the relatively minor problem of data access incompatibility when a newer version of software replaces an older version of software. On the other hand, an older version of software generally cannot handle the data format and memory configuration used by later versions of software. Thus, in situations like those described above where an older version of software replaces a newer version of software, manual intervention must occur at the site of data memory 17 in order to manually reformat the data and reconfigure the memory for compatibility with the data handler of the older version of software that has been downloaded.

Another conventional approach is to update the data handler of every existing version of software whenever a new version of software with a different data handler and different data format and data configuration requirements is released. This is of course very costly and time consuming for the software vendor. Moreover, it is difficult to get such updated software into use in all customer systems. The customer

does not want to "disturb" a system more than absolutely needed, which may force the supplier to provide system components which each have a different software version pre-programmed therein but are otherwise identical. The supplier must then keep track of which software version is used by which customer, and deliver equipment with that customer's software version programmed therein in order to avoid the above-described process of replacing a newer software version with an older software version when the equipment is installed in the system. This requirement of providing various equipments with different software versions therein and keeping track of which software versions are used by which customer systems adds more complexity to production of the equipment, which is very costly and time consuming for the supplier.

It is therefore desirable to provide for remote downloading of an older version of software to replace a newer version of software wherein the data format and memory configuration provided to the older version of software is compatible with the older version of software, without need for manual intervention at the site of the data memory, without need for updating the data handler portion of every software version whenever a new software version is released, and without need to design the earliest software version such that its data handler will be compatible with the data handlers of every subsequent future version of the software.

The present invention provides for remote downloading of an older version of software to replace a newer version of software wherein the downloaded older version of software encounters a data format and memory configuration with which it is compatible. The technique of the present invention does not require manual intervention at the site of the data memory, does not require updating the data handler of every existing version of software whenever a new version of software is released, and does not require the data handler of the first version of software to meet all data handling, data formatting and memory configuration needs of every subsequent version of the software released in the future.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 illustrates a conventional distributed computer system wherein software from a source located remotely away from a data processor is downloaded into a program memory of the data processor.

FIGURE 2 illustrates conceptually the effect of downloading software from the remote source into the data processor of FIGURE 1.

FIGURE 3 illustrates the data handler portion and accessed data associated with the current software version shown in FIGURE 2.

FIGURE 4 illustrates the data handler portion and accessed data associated with the downloaded software version shown in FIGURE 2.

FIGURE 5 is a flow diagram of a conventional download control portion of the current software version of FIGURE 2.

FIGURE 6 is a flow diagram of a conventional download reformat portion of the downloaded software version illustrated in FIGURE 2.

FIGURE 7 is a flow diagram of a download control portion to be executed by the current software version according to the present invention.

FIGURE 8 is a flow diagram of a download reformat portion to be executed by the downloaded software according to the present invention.

FIGURE 9 illustrates pertinent portions of all new software versions according to the present invention.

FIGURE 10 illustrates diagrammatically a cellular telephone system incorporating the software download techniques of the present invention.

FIGURE 11 illustrates one of the base station controllers and one of the base transceiver stations of FIGURE 10 in greater detail.

DETAILED DESCRIPTION

Example FIGURE 7 illustrates a download control routine 70 to be executed by the currently executing software SWc. according to the present invention. At 71, the downloaded software SWd is received and stored. At 73, SWc waits for a command from the remote source 11 to begin execution of SWd. When the command to begin execution of SWd is received, SWc determines at 75 whether or not SWd is an older software version than is SWc. This can be determined, for example, by comparing a status field in SWd with a corresponding status field in SWc.

If SWd is not an older version than SWc, then SWc at 77 causes the data processor to restart, running SWd. If the downloaded software SWd is older than the currently running software SWc, then SWc, by virtue of being a later version than

SWd, determines at 79 whether or not the database is supported by SWd. If the database is not supported by SWd, then SWc reformats the database at 72. At 74, SWc determines whether or not SWd supports the current memory configuration. If SWd does not support the current memory configuration, then SWc at 76 changes the memory configuration to be compatible with SWd. Thus, SWc reformats the database only if necessary for compatibility with SWd, and SWc changes the current memory configuration only if necessary for compatibility with SWd. If the current database is compatible with SWd, then it remains unchanged, and if the current memory configuration is compatible with SWd, then it remains unchanged.

Example FIGURE 8 illustrates the download reformat portion 80 of the downloaded software SWd, according to the present invention. When SWc causes the data processor 13 to restart running SWd at 77 in FIGURE 7, SWd then determines at 81 whether or not SWd is an older version than SWc. If so, then any necessary data reformatting and memory reconfiguration have already occurred at 72 and 76 of the download control portion of SWc in FIGURE 7. If SWd is not older than SWc at 81, then SWd is a newer version than SWc. Thus, SWd determines at 83 whether or not it supports the current database format, and reformats as needed at 85, and then determines at 87 whether or not it supports the current memory configuration, and changes the configuration as needed at 89.

It can be seen from the foregoing description of FIGURES 7 and 8 that the combination of download control routine 70 of SWc and download reformat routine 80 of SWd provide for data format and memory configuration compatibility regardless of whether the downloaded software SWd is a newer version or an older version than currently executing software SWc.

Example FIGURE 9 illustrates another feature of the present invention wherein all new software versions S_{NEW} include download control routine 70 from FIGURE 7 and download reformat routine 80 from FIGURE 8, along with the unique data handler portion 91 associated with that particular version of software (and other unillustrated functional portions that are not necessary to understand the invention). With the download control 70 and download reformat 80 included in all software versions, whenever any software version is downloaded to replace any other software version, and regardless of which version is older, the download process will automatically

render the data format and memory configuration compatible to the requirements of data handler 91. The currently executing software will execute the download control routine 70, while the downloaded software will execute the download reformat routine 80. Thus, the data processor will access data memory as needed to render its configuration and data format compatible with data handler 91.

Although data format and memory configuration are targeted for adjustment in the examples of FIGURES 7 and 8, any desired data access parameter can be adjusted using the procedures of FIGURES 7 and 8. For example, any desired target parameter could be considered at 83 or 79, and adjusted as needed at 85 or 72.

Also, data handler 91 of each new software release S_{NEW} advantageously includes full knowledge of the manner in which all data handlers of previous versions handle data (e.g., data formats, memory configurations). Such information is readily available and easily included in new software releases. This information permits the software S_{NEW} to determine the need for, and perform as needed, the reformat and reconfiguration functions shown in FIGURES 7 and 8.

Example FIGURE 10 illustrates a cellular telephone system which incorporates the above-described features of the present invention. As is conventional, a public switched telephone network 101 is connected to a mobile services switching center 103, which is in turn connected to base station controllers 105, which are in turn connected to base transceiver stations 107. The base transceiver stations then are connected via the air interface to mobile communication devices such as mobile telephones at 109.

Example FIGURE 11 illustrates one of the base station controllers 105 of FIGURE 10 executing a remote software download to a program memory 111 of a data processor 113 in one of the base transceiver stations 107. The base transceiver station (BTS) at 107 also includes data memory 117 connected to data processor 113, and data processor 113 includes execute portion 115 to execute software stored in program memory 111.

It should be noted that the "data" that is handled and reformatted by SWc and SWd could be pure data or a program such as a library routine, and "data" memories 17 and 117 could hold pure data and/or one or more programs.

Some exemplary reasons why data in a database may need to be reformatted include: (1) to add a database element/parameter (with some default values) that was removed by SWc but is required by SWd; (2) to remove a database element/parameter that was added by SWc but is not supported by SWd; and (3) to change the size of a database element/parameter to a size supported by SWd, for example, the size of an error log may have been increased or decreased from one software release to another.

Some exemplary reasons for changing the memory configuration include: (1) to change the memory location of a database element/parameter such as an error log; (2) to change the memory location of a program stored in the memory; and (3) to change the storage structure of programs such as library routines that are stored in the memory, for example, changing between a "stacked" storage structure (wherein the software modules are stored in memory consecutively, one after another) and a "segmented" storage structure (wherein the software is divided into small segments and stored in the next free segment).

Using the present invention as described above, there are more possibilities to introduce new functions into a distributed system with less concern for forward and backward software capability. Also, the above-described disadvantages associated with the conventional solutions are avoided because all that is necessary according to the invention is to include the download control routine 70 and the reformat control routine 80 in each new software version, which is a virtually negligible expense in terms of time and money.

Although exemplary embodiments of the present invention have been described above in detail, this does not limit the scope of the invention, which can be practiced in a variety of embodiments.

WHAT IS CLAIMED IS:

1. A method of replacing a first version of software with a second version of software, comprising:

running the first version of software on a data processor;

loading the second version of software into the data processor; and

using the first version of software to determine whether the second version of software is an earlier version than the first version of software.

2. The method of Claim 1, including using the first version of software to modify a memory for compatibility with the second version of software if the second version of software is an earlier version than the first version of software.

3. The method of Claim 2, wherein the first version of software modifies one of (a) the format of data in the memory and (b) the memory configuration of the memory.

4. The method of Claim 2, including selectively using the second version of software to modify the memory for compatibility with the second version of software if the first version of software is an earlier version than the second version of software.

5. The method of Claim 1, including selectively using the second version of software to modify a memory for compatibility with the second version of software if the first version of software is an earlier version than the second version of software.

6. The method of Claim 1, wherein the data processor is provided in a base transceiver station of a cellular telephone system, and wherein the second version of software is loaded into the data processor from a base station controller of the cellular telephone system.

7. The method of Claim 1, wherein said loading step includes downloading the second version of software from a source apparatus into the data processor via a communication path that exists between the data processor and the source apparatus and that is also used for other communications between the source apparatus and the data processor.

8. A method of replacing a first version of software with a second version of software, comprising:

running the first version of software on a data processor;

loading the second version of software into the data processor; and

using the first version of software to modify a memory for compatibility with the second version of software.

9. The method of Claim 8, including performing said step of using the first version of software only if the second version of software is an earlier version than the first version of software.

10. The method of Claim 8, wherein said step of loading the second version of software includes downloading the second version of software from a source apparatus into the data processor via a communication path that exists between the data processor and the source apparatus and that is also used for other communications between the source apparatus and the data processor.

11. The method of Claim 8, wherein the data processor and memory are provided in a base transceiver station of a cellular telephone system, and wherein the second version of software is loaded into the data processor from a base station controller of the cellular telephone system.

12. The method of Claim 8, wherein the first version of software modifies one of (a) the format of data in the memory and (b) the memory configuration of the memory.

13. A data processing system, comprising:

a memory;

a data processor connected to said memory, said data processor including a program memory and an execution portion for executing a first version of software stored in said program memory, said data processor having an input for receiving a second version of software; and

said data processor, while executing said first version of software, responsive to receipt of the second version of software at said input for using said first version of software to access and modify said memory for compatibility with said second version of software.

14. The system of Claim 13, including a source apparatus for providing said second version of software, and a communication path coupling said source apparatus to said input of said data processor, wherein said communication path carries said second version of software from said source apparatus to said data

processor, and wherein said communication path also carries between said source apparatus and said data processor information other than software.

15. The system of Claim 13, wherein said data processing system is a cellular telephone system, and wherein said memory and said data processor are provided in a base transceiver station of said cellular telephone system, and including a base station controller connected to said input of said data processor for providing said second version of software to said input of said data processor.

16. The system of Claim 13, wherein said data processor uses said first version of software to access and modify said memory only if said second version of software is an earlier version than said first version of software.

17. The system of Claim 13, wherein said data processor modifies one of (a) the format of data in the memory and (b) the memory configuration of the memory.

18. The system of Claim 14, wherein said communication path includes a communication bus.

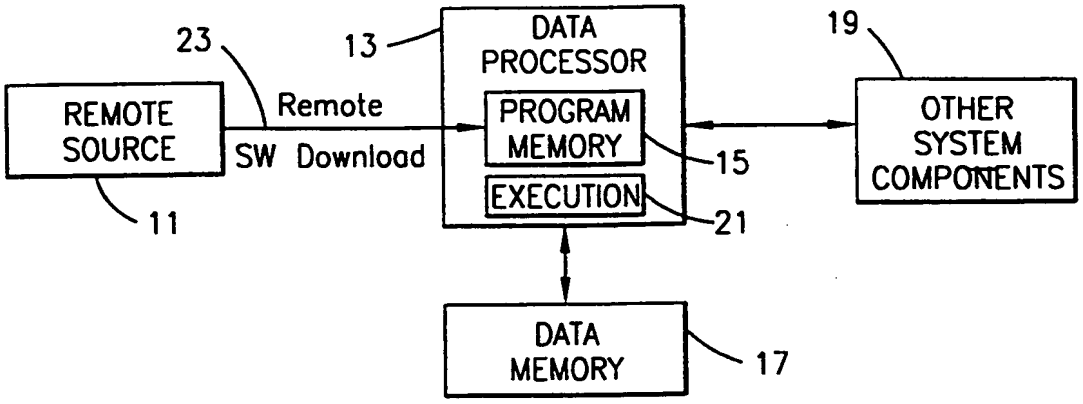

FIG. 1
(PRIOR ART)

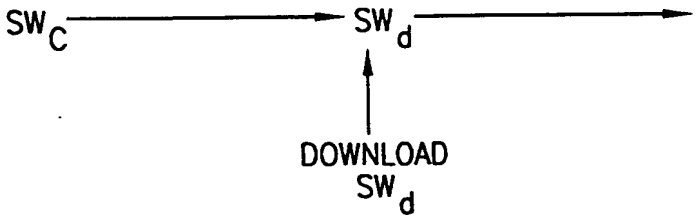

FIG. 2
(PRIOR ART)

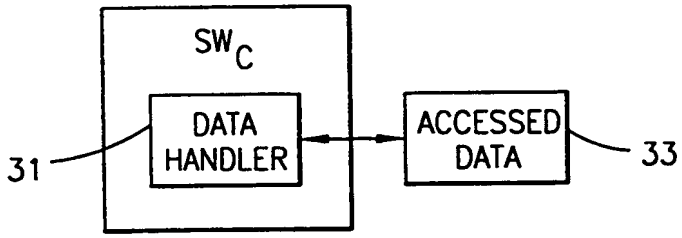

FIG. 3
(PRIOR ART)

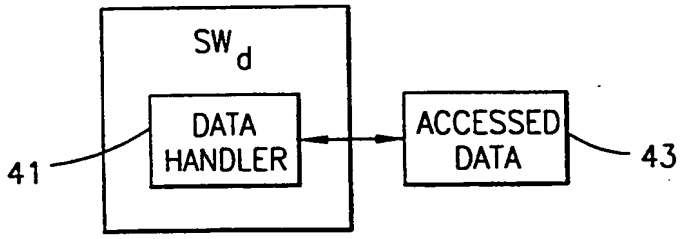

FIG. 4
(PRIOR ART)

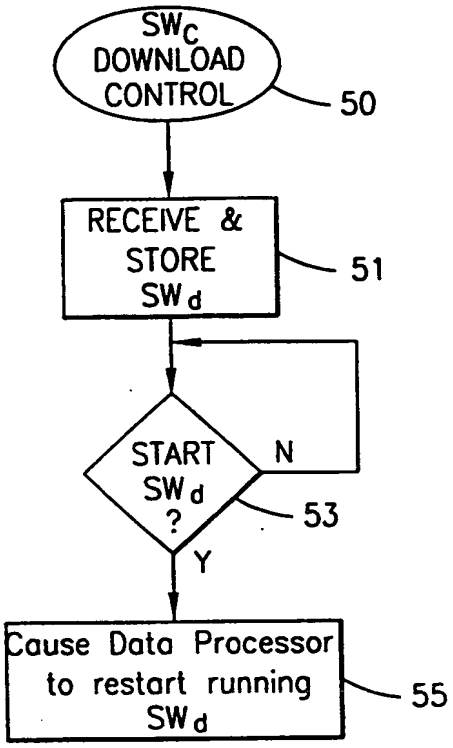

FIG. 5
(PRIOR ART)

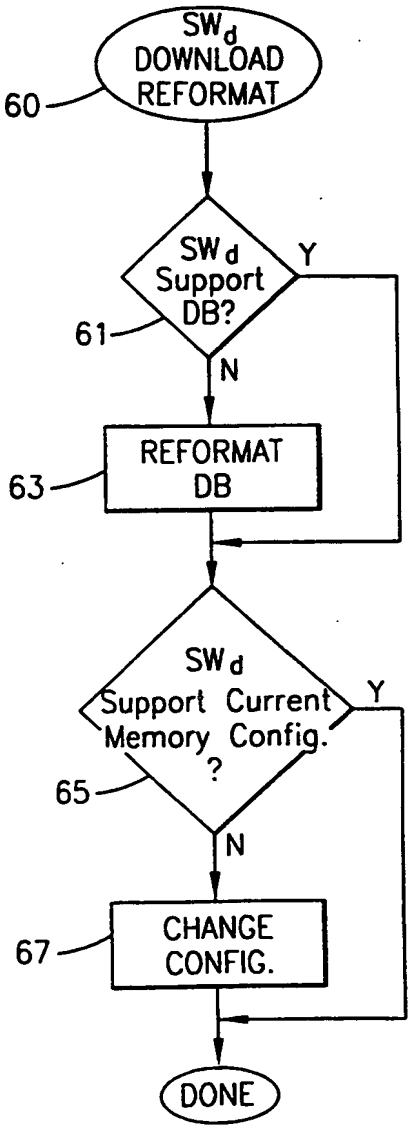

FIG. 6
(PRIOR ART)

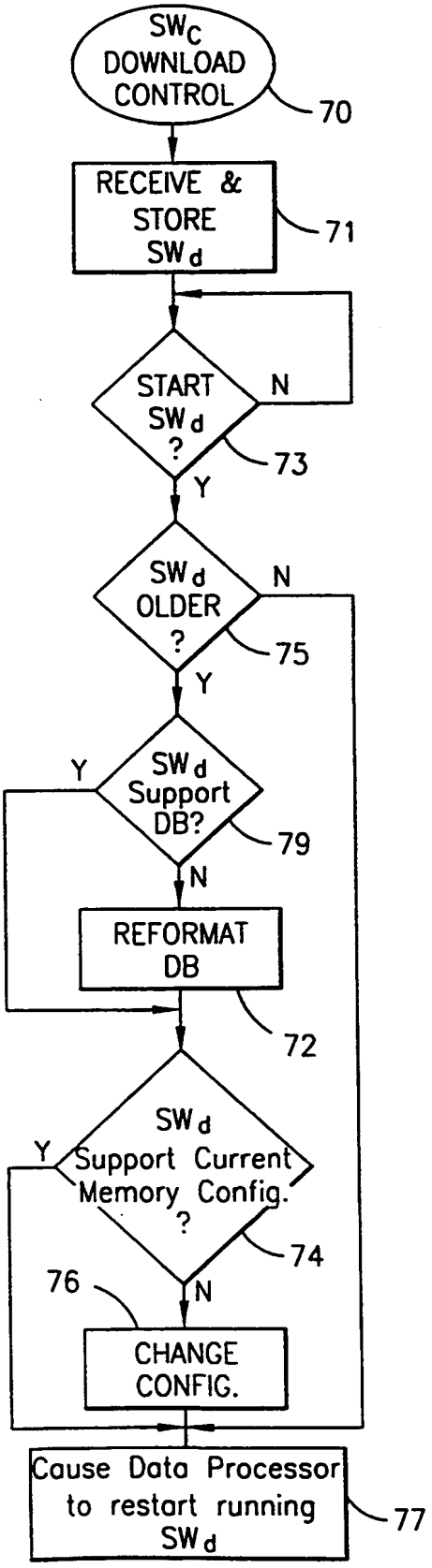
SWC DOWNLOAD CONTROL
70
RECEIVE & STORE SWd
71
START SWd ?
N
73
Y
SWd OLDER ?
N
75
Y
SWd Support DB?
Y
79
N
REFORMAT DB
72
SWd Support Current Memory Config. ?
Y
74
N
76
CHANGE CONFIG.
Cause Data Processor to restart running SWd
77

FIG. 7

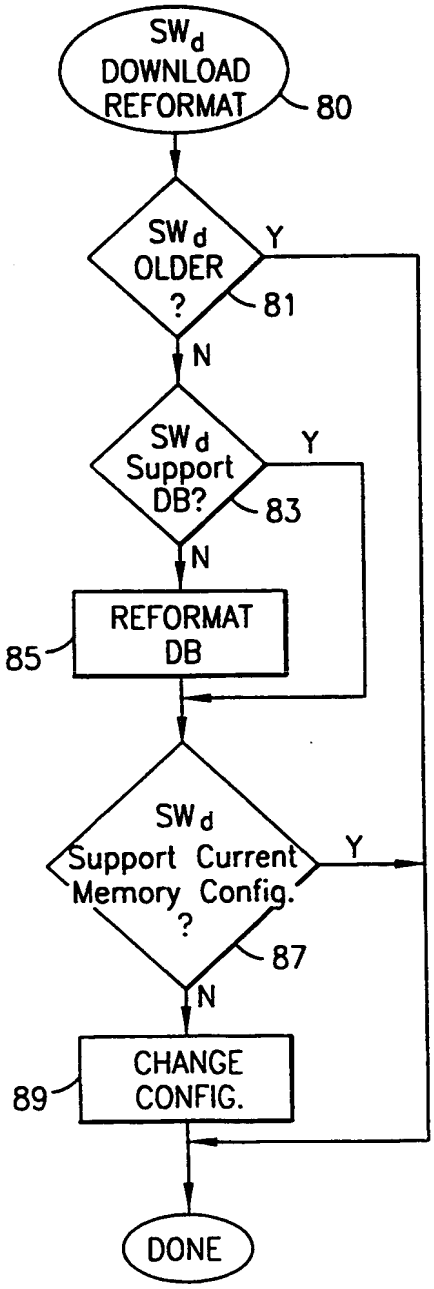

FIG. 8

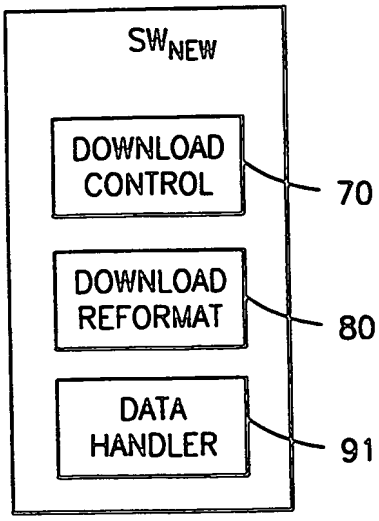

FIG. 9

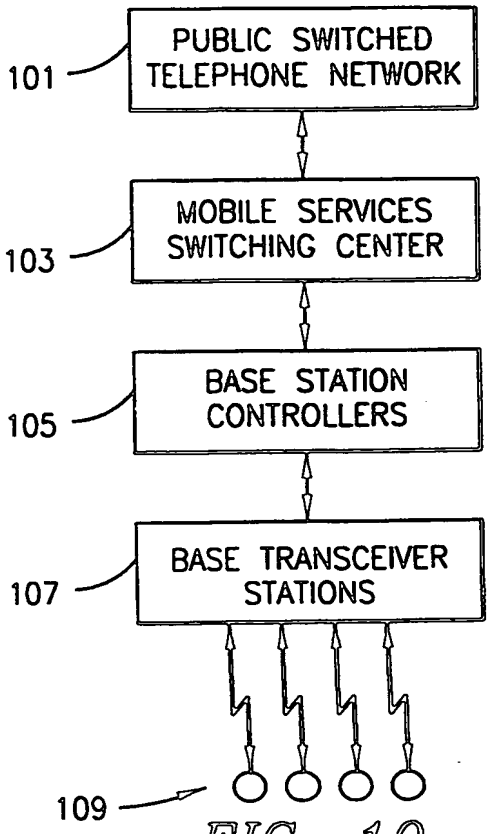

FIG. 10

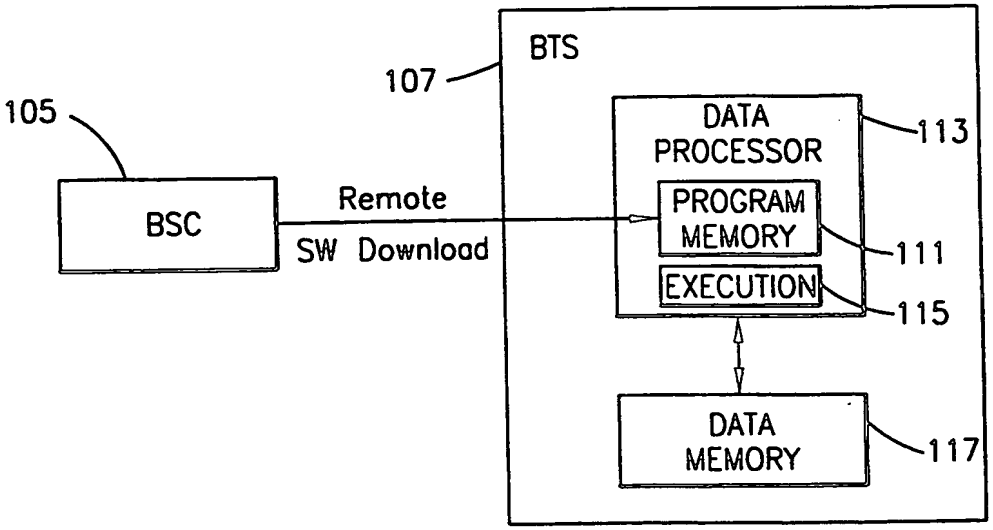

FIG. 11

INTERNATIONAL SEARCH REPORT

Interi ial Application No
PCT/SE 98/01943

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/445 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 632 371 A (XEROX CORP) 4 January 1995 see abstract see page 5, line 48 - page 6, line 5	1,8,13
A	US 5 410 703 A (NILSSON RICKARD ET AL) 25 April 1995 see the whole document	1,8,13
A	"VERSION EQUALIZER PROGRAM FOR SERVER/CLIENT" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 38, no. 8, 1 August 1995, page 575/576 XP000534636	1,8,13
A	EP 0 498 130 A (IBM) 12 August 1992 see the whole document	1,8,13

☐ Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 December 1998	23/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Fonderson, A

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

....ormation on patent family members

Interr ıal Application No
PCT/SE 98/01943

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0632371 A	04-01-1995	BR 9402027 A	13-12-1994
		JP 7006026 A	10-01-1995
		US 5499357 A	12-03-1996
US 5410703 A	25-04-1995	AU 667559 B	28-03-1996
		AU 4516493 A	31-01-1994
		CN 1081010 A	19-01-1994
		EP 0648353 A	19-04-1995
		FI 946195 A	30-12-1994
		MX 9303648 A	31-01-1994
		NO 945096 A	22-02-1995
		WO 9401819 A	20-01-1994
		US 5555418 A	10-09-1996
EP 0498130 A	12-08-1992	US 5579509 A	26-11-1996
		JP 5088859 A	09-04-1993